Linear Algebra and its Applications 436 (2012) 1267–1284

Contents lists available at SciVerse ScienceDirect

Linear Algebra and its Applications

journal homepage: www.elsevier.com/locate/laa

Combining a hybrid preconditioner and a optimal adjustment algorithm to accelerate the convergence of interior point methods

Carla T.L.S. Ghidini [a,*], A.R.L. Oliveira [a], Jair Silva [b], M.I. Velazco [c]

[a] *Institute of Mathematics, Statistics and Scientific of Computation (IMECC), University of Campinas (UNICAMP), C.P. 6065, 13.083-970 Campinas, SP, Brazil*
[b] *Mathematics Department, Federal University of Mato Grosso do Sul, C.P. 549, 79070-900 Mato Grosso do Sul, MS, Brazil*
[c] *Campo Limpo Paulista School, 13.231-230 Campo Limpo Paulista, SP, Brazil*

ARTICLE INFO

Article history:
Received 4 April 2011
Accepted 29 July 2011
Available online 9 September 2011

Submitted by V. Mehrmann

Keywords:
Linear programming
Preconditioning
Interior point methods

ABSTRACT

In this work, the optimal adjustment algorithm for p coordinates, which arose from a generalization of the optimal pair adjustment algorithm is used to accelerate the convergence of interior point methods using a hybrid iterative approach for solving the linear systems of the interior point method. Its main advantages are simplicity and fast initial convergence. At each interior point iteration, the preconditioned conjugate gradient method is used in order to solve the normal equation system. The controlled Cholesky factorization is adopted as the preconditioner in the first outer iterations and the splitting preconditioner is adopted in the final outer iterations. The optimal adjustment algorithm is applied in the preconditioner transition in order to improve both speed and robustness. Numerical experiments on a set of linear programming problems showed that this approach reduces the total number of interior point iterations and running time for some classes of problems. Furthermore, some problems were solved only when the optimal adjustment algorithm for p coordinates was used in the change of preconditioners.

1. Introduction

Interior point methods have been the object of intensive research since the appearance of their first polynomial version in the 80's. Their good practical performance and theoretical properties have

* Corresponding author.
E-mail addresses: carla@ime.unicamp.br (C.T.L.S. Ghidini), aurelio@ime.unicamp.br (A.R.L. Oliveira), jairmt@gmail.com (J. Silva), marta.velazco@gmail.com (M.I. Velazco).

0024-3795/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.laa.2011.08.023

motivated the implementation of sophisticated codes to solve large scale linear programming problems. The success of these methods rely mainly on the fact that the convergence is achieved in relatively few iterations [1].

Each iteration of an interior point method involves the solution of one or more linear system [2–4]. This is the most expensive step of these methods. There are several approaches for solving the linear systems. Usually, direct methods are adopted by reducing the indefinite augmented system to the normal equation one and applying the Cholesky factorization [5,6,2,3]. However, this approach cannot be applied for some classes of large scale problems due to memory and/or time limitations. For these problems, the iterative method for the solution of the linear system would be the chosen approach [7–10].

In this work, an iterative hybrid approach is used to solve the normal equation system that arises in an interior point method for linear programming. The conjugate gradient method is preconditioned during the initial interior point iterations using a kind of incomplete factorization called controlled Cholesky factorization [11], and in the remaining iterations, when these systems become highly ill-conditioned, using a specially tailored preconditioner, the splitting preconditioner developed in [10].

The transition between both preconditioners is critical in the sense that if it happens too early, the splitting preconditioner is not yet fit for the job. However, if it happens too late, the controlled Cholesky factorization preconditioner is no longer effective [12]. In this situation, the method would fail.

On the other hand, a new trend in the past few years consists in the use of simple linear programming methods in order to give a warm starting point for interior point methods, reducing the total number of iterations [13]. The von Neumann's algorithm is one of the first used in such applications since its iteration is very cheap and it has fast initial convergence.

In order to deal with this problem, we perform a few iterations of the optimal adjustment algorithm for p coordinates, between some interior point method iterations in order to improve the solution. In particular, it is applied just before the change of preconditioners, to improve the current point and to deliver a point closer to an optimal solution for the splitting preconditioner. This approach close the gap in the transition of preconditioners for some tested problems.

The optimal adjustment algorithm for p coordinates was proposed in [14] and it is a generalization of the optimal pair adjustment algorithm developed in [13], which is based on the classical von Neumann's algorithm. It has interesting properties such as simplicity and fast initial convergence, which motivated its use.

Numerical experiments with large scale linear programming problems show that such strategy allows to improve performance, achieving both faster convergence and adding robustness to the whole approach.

The paper is organized as follows. In Section 2, the primal–dual interior point methods are briefly reviewed and the linear systems that need to be solved are presented. In Section 3, the hybrid preconditioner, controlled Cholesky factorization and splitting preconditioner are presented and some features of these preconditioners are discussed. In Section 4, simple algorithms for linear programming such as the von Neumann's algorithm, the optimal pair adjustment algorithm and the optimal adjustment algorithm for p coordinates are described and several theoretical properties are discussed. Section 5 describes the computational experiments. In Section 6, conclusions are drawn and future perspectives are suggested.

2. Primal–dual interior-point methods

Consider the linear programming problem in the standard form,

$$\begin{aligned} \text{Min } & c^T x \\ \text{s.t. } & Ax = b \\ & x \geqslant 0, \end{aligned} \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$ and $m \leqslant n$. The dual problem associated with problem (1) is

$$\begin{aligned} &\text{Max } b^T y \\ &\text{s.t. } A^T y + z = c \\ &\qquad z \geqslant 0, \end{aligned} \tag{2}$$

where $y \in \mathbb{R}^m$ is a column vector of free variables and $z \in \mathbb{R}^n$ is a column vector of dual slack variables.

The Karush–Kuhn–Tucker optimality conditions for (1) and (2) are

$$\begin{aligned} &Ax - b = 0 \\ &A^T y + z - c = 0 \\ &XZe = 0 \\ &(x, z) \geqslant 0, \end{aligned} \tag{3}$$

where $e \in \mathbb{R}^n$ is the vector of all ones and $X = diag(x)$, $Z = diag(z)$.

The solution for this problem can be obtained by solving the non-linear system (3). If (x, y, z) is a solution of (3), then x and (y, z) are optimal solutions of the problems (1) and (2), respectively.

The gap for the problem (1) is given by $\gamma = c^T x - b^T y$. However, it can be reduced to $\gamma = x^T z$ for a feasible primal and dual point.

Remark 2.1. Problems with bounded variables will not be subject of study in this section. Nevertheless, the following discussion can be easily extended when bounded variables are considered and the implementation described in the numerical results considers these variables.

The solution of (3) can be determined using primal–dual methods, which are based on Newton's method applied to the optimality conditions without the non-negativity of the (x, z) components. Mehrotra's predictor–corrector method [4] is one of the most successful among the interior-point methods. The search direction is computed by solving two linear systems, which have the same coefficient matrix, but different right-hand sides. The affine-scaling direction is computed as follows:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I_m \\ Z^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta_a x \\ \Delta_a y \\ \Delta_a z \end{bmatrix} = \begin{bmatrix} r_p^k \\ r_d^k \\ r_a^k \end{bmatrix} \tag{4}$$

where $r_p^k = b - Ax^k$, $r_d^k = c - A^T y^k - z^k$ and $r_a^k = -X^k Z^k e$.

To obtain the centering corrector direction $(\Delta_c x^k, \Delta_c y^k, \Delta_c z^k)$, the right-hand side vector of (4) is set to $r_d^k = 0$, $r_p^k = 0$ and $r_a^k = \mu_k e - \Delta_a X^k \Delta_a Z^k e$ where μ_k is the centering parameter, $\Delta_a X^k = \text{diag}(\Delta_a x^k)$ and $\Delta_a Z^k = \text{diag}(\Delta_a z^k)$.

Then the search direction (Δ) is determined as follows: $\Delta = \Delta_a + \Delta_c$.

To avoid this addition, the search directions may be computed solving only the system (4) with r_a^k set to $\mu_k e - \Delta_a X^k \Delta_a Z^k e - X^k Z^k e$.

2.1. Linear system solution

In this section, only the linear system (4) will be considered, because both predictor–corrector systems share the same coefficient matrix. This system may be reduced to an augmented indefinite linear system by eliminating the variables Δz from the second equation. Defining $D = Z^{-1}X$, the augmented system that has a symmetric form is the following:

$$\begin{bmatrix} -D^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_d - X^{-1} r_a \\ r_p \end{bmatrix} \tag{5}$$

Matrix A is full rank. Then system (5) can be reduced to a smaller positive definite normal equation system by eliminating Δx from the first equation,

$$(ADA^T)\Delta y = AD(r_d - X^{-1} r_a) + r_p. \tag{6}$$

Both direct and iterative methods can be applied to solve either system (5) or (6). The Cholesky factorization, which has the advantage of working with symmetric positive definite matrices, is the most common approach used in interior-point methods to solve the normal equation system. However, it can be very expensive if the Cholesky factor is dense. The presence of few dense columns in A lead to loss of sparsity in ADA^T. In this case, the use of iterative methods becomes interesting. A classical iterative method used to solve normal equation systems is the preconditioned conjugate gradient method. However, to build the preconditioner it is often necessary to compute ADA^T, which is usually much less sparse than A. Moreover, it is in general more difficult to find a good preconditioner to the normal equation matrix than to the augmented one.

For this reason, the augmented system strategy has been considered in several papers [7,9,15,16,10] even though it is indefinite. The Cholesky factorization cannot be applied since there is no numerically stable way to factor a general indefinite matrix onto LDL^T. An alternative used in interior-point methods is changing the indefinite system to a quasidefinite and applying the primal and dual regularization method [17]. In this case, the Cholesky-like factorization LDL^T with a diagonal D exists for any symmetric row and column permutation of the matrix.

It is not guaranteed that the conjugate gradient method will achieve convergence when the system is indefinite. However, it can be applied with a convenient preconditioner [18,15].

Most preconditioners for indefinite systems from interior-point methods are developed for solving non-linear and quadratic programming problems and almost always they are not efficient in solving large-scale linear programming problems. However, a class of preconditioners called splitting preconditioners was designed especially for indefinite systems arising from linear programming problems [10]. An important feature of this class is the option to reduce the preconditioned indefinite system to a positive definite one like the normal equation system allowing for the application of the conjugate gradient method. Moreover, the splitting preconditioners have better results near of the optimal solution. Therefore, an alternative preconditioner is necessary for the initial interior-point iterations.

3. Preconditioner

In this section, the splitting preconditioner, developed by Oliveira and Sorensen [10] and the controlled Cholesky factorization (CCF) preconditioner built by Campos and Birkett [11,19] are briefly described. Both preconditioners are used by a hybrid approach of preconditioning proposed by Bocanegra et al. [8].

3.1. The splitting preconditioner

The splitting preconditioner, proposed in [10], is a generalization of the preconditioner proposed by Resende and Veiga [20] in the context of the minimum cost network flow problem. The main feature of this class is that it works better near a solution of the linear programming problem. This is an advantage as the linear system is known to be very ill-conditioned close to a solution and these systems are difficult to solve using iterative methods. Additionally, the splitting preconditioner avoids the normal equation computation. However, since the preconditioner is especially tailored for the final iterations of the interior point methods, it fails to obtain convergence in the initial iterations for many linear programming problems.

The splitting preconditioner is described as follows:

Let $A = [B\ N]P$ where $P \in \mathbb{R}^{n\times n}$ is a permutation matrix such that $B \in \mathbb{R}^{m\times m}$ is nonsingular ($N \in \mathbb{R}^{m\times(n-m)}$), then

$$ADA^T = [B\ N]PDP^T \begin{bmatrix} B^T \\ N^T \end{bmatrix} = [B\ N] \begin{bmatrix} D_B & 0 \\ 0 & D_N \end{bmatrix} \begin{bmatrix} B^T \\ N^T \end{bmatrix} = BD_BB^T + ND_NN^T\ .$$

The preconditioner is given by $D_B^{-\frac{1}{2}}B^{-1}$ and the preconditioner matrix M is as follows:

$$M = D_B^{-\frac{1}{2}}B^{-1}(ADA^T)B^{-T}D_B^{-\frac{1}{2}} = I_m + GG^T, \tag{7}$$

where $G = D_B^{-\frac{1}{2}}B^{-1}ND_N^{\frac{1}{2}}$.

The product $B^{-1}N$ can be seen as a scaling of the linear programming problem. Close to a solution, at least $n-m$ entries of D are small. Thus, with a suitable choice of the B columns, the diagonal entries of D_B^{-1} and D_N are very small close to a solution. In this situation, G approaches the zero matrix, M approaches the identity matrix and both the largest eigenvalue of M and $\kappa_2(M)$ approach one.

A nice property of the splitting preconditioner is that it can work with the selected set of columns for some iterations. As a consequence, the preconditioner is very cheap to compute for these iterations. It is important to notice that keeping the matrix B from previous iterations does not mean keeping the same preconditioner since it will change from an iteration to the next following the changes of diagonal scaling matrix D.

A detailed description on how to implement the splitting preconditioner efficiently can be found in [21].

3.1.1. Column ordering

The price paid for avoiding the normal equation system is the need to find B and solve linear systems using it. However, the factorization $QB = LU$, where Q is a permutation matrix, is typically easier to compute than the Cholesky factorization [22].

A strategy to form B is to minimize $||G||$ since close to a solution, the preconditioned matrix approaches the identity for a suitable choice of B columns which aims at keeping both D_B^{-1} and D_N small. This problem is hard to solve. An approximate solution was obtained in [10] by selecting the first m linearly independent columns of AD with a smallest norm-1 to form B.

However, the 1-norm has a tendency to diminish the effect of outliers, which is an undesirable feature in this context as the aim is to split the columns into two sets of size m and $n-m$, respectively. In [12], the 2-norm was used instead of the 1-norm, trying to avoid this tendency. This improved the performance of the splitting preconditioner for most problems and allowed for better computational results and reduced the number of iterations for the convergence of the conjugate gradient method.

Therefore, in this work the 2-norm is used to select the columns of AD.

3.2. Controlled Cholesky factorization preconditioner

The controlled Cholesky factorization (CCF) preconditioner, designed for solving general positive definite systems [11], can be seen as a variation of the incomplete Cholesky factorization. The main objective of this factorization is to build a preconditioned matrix that has grouped eigenvalues and which is near the unit in order to accelerate the convergence of the conjugate gradient method.

Cholesky factorization of the matrix $PADA^TP^T \in \mathbb{R}^{m\times m}$ is as follows:

$$PADA^TP^T = LL^T = \bar{L}\bar{L}^T + R,$$

where P is a symmetric pivoting of ADA^T for sparseness, L is the factor obtained when factorization is complete, $\bar{L}$ is the factor obtained when factorization is incomplete and R is a remainder matrix.

Using $\bar{L}$ as a preconditioner matrix for ADA^T,

$$\bar{L}^{-1}(ADA^T)\bar{L}^{-T} = (\bar{L}^{-1}L)(L^T\bar{L}^{-T}) = (\bar{L}^{-1}L)(\bar{L}^{-1}L)^T$$

Let$F = L - \bar{L}$. Replacing F in the last equality

$$\bar{L}^{-1}(ADA^T)\bar{L}^{-T} = (I_m + \bar{L}^{-1}F)(I_m + \bar{L}^{-1}F)^T.$$

Note that when $\bar{L} \approx L$ then $F \to 0$ and, therefore, $\bar{L}^{-1}(ADA^T)\bar{L}^{-T} \to I_m$.

Duff and Meurant [23] showed that the number of iterations needed for convergence by the conjugate gradient method is directly related to the norm of R, which is $R = LF^T + FL^T - FF^T$.

It is assumed that matrix ADA^T has been diagonally scaled to give a unit diagonal [24] in order to improve robustness.

The CCF is based on the minimization of the Frobenius norm of F. Therefore, when $||F|| \to 0 \Longrightarrow ||R|| \to 0$.

Consider the following problem:

$$\text{Min } \|F\|_F^2 = \sum_{j=1}^{m} c_j \quad \text{with} \quad c_j = \sum_{i=1}^{m} |l_{ij} - \bar{l}_{ij}|^2.$$

where l_{ij} are the element of L.

Splitting c_j in two summations:

$$c_j = \sum_{k=1}^{t_j+\eta} |l_{i_k j} - \bar{l}_{i_k j}|^2 + \sum_{k=t_j+\eta+1}^{m} |l_{i_k j}|^2,$$

where t_j is the number of nonzero entries below the diagonal in the jth column of matrix ADA^T and η is the number of extra entries allowed per column in the incomplete factorization.

The first summation contains all $t_j + \eta$ nonzero entries of the jth column of $\bar{L}$. The second one has only the remaining entries of the complete factor L which do not have the corresponding entries in $\bar{L}$.

Thus, the problem can be solved using the following heuristic:

- Increasing η (allowing more fill-in). The term c_j should decrease because the first summation contains more elements.
- Choosing the $t_j + \eta$ largest entries of $\bar{L}$ in an absolute value for fixed η. In this case, the largest entries are in the first summation leaving only the smallest l_{ij} in the second summation, producing an optimal factor $\bar{L}$.

The preconditioner $\bar{L}$ is built by columns. Consequently, it needs only the jth column of ADA^T at each time.

The main features of the CCF preconditioner are: choice of entries by value; avoiding loss of positive definiteness by exponential shift; versatile preconditioner (the number of nonzero entries per column can vary from 1 to m); predictable storage.

Jones and Plassmann [25] developed an approach which can have a fixed number of nonzero entries in each row or column of the preconditioner. It is taken to be the number of nonzero entries in each row or column of the original coefficient matrix. Nevertheless, only the largest entries in magnitude are kept, meaning that the original sparsity pattern is ignored. Thus CCF can be seen as a generalization of the Jones and Plassmann method since fill-in is allowed in CCF.

For more details of the CCF preconditioner see [8].

3.3. Hybrid preconditioner

The matrix D changes significantly from one interior-point iteration to the next and it becomes highly ill-conditioned in the final iterations. For this reason, it is difficult to find a preconditioning strategy that has a good performance of iterative methods over the entire course of the interior-point iterations.

In [8] it was proposed to apply the conjugate gradient method to solve system (6) preconditioned by a hybrid preconditioner matrix M,

$$M^{-1}(ADA^T)M^{-T}\bar{y} = M^{-1}(AD(r_d - X^{-1}r_a) + r_p), \tag{8}$$

where $\bar{y} = M^T\Delta y$.

This approach assumes the existence of two phases during interior-point iterations. In the first one, the controlled Cholesky preconditioner is used to build matrix M. After the change of phases, matrix M is built using the splitting preconditioner.

3.3.1. Change of phases

In [8] the change of phase happens when the initial gap $(x_0^T z_0)$ for the linear programming problem is reduced by a factor of 10^{-6} or the number of inner iterations to solve the linear system reaches $m/2$, where m is the dimension of ADA^T.

In [12], a new heuristic for change of preconditioners was proposed. If the number of iterations needed for the conjugate gradient method to achieve convergence is greater than $\frac{m}{6}$, the parameter η in the controlled Cholesky factorization is increased, i.e., $\eta = \eta + 10$. The change occurs when η exceeds a fixed maximum η. In this work, a change of phase happens when the number of iterations needed for the conjugate gradient method to achieve convergence is greater than $\frac{m}{6}$ and when the parameter η in the controlled Cholesky factorization falls above the maximum allowed.

However, this approach can fail to achieve convergence for some classes of linear programming problems when the controlled Cholesky factorization is not longer effective and at the same time, the splitting preconditioner is not yet prepared for the job.

4. Simple algorithms for linear programming

Consider the problem of finding a feasible solution of the following set of linear constraints:

$$\begin{aligned} &Ax = 0,\\ &e^T x = 1,\\ &x \geqslant 0, \end{aligned} \tag{9}$$

where $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$, $e \in \mathbb{R}^n$ is the vector of all ones, and the columns of A have norm one, i.e., $||A_j|| = 1$, for $j = 1, \ldots, n$.

Geometrically, columns A_j can be viewed as points lying on the m-dimensional hypersphere with unit radius and a center at the origin. This problem can be described as assigning non-negative weights x_j to columns A_j in such a way that after being re-scaled its center of gravity is the origin.

Remark 4.1. Any linear programming problem can be reduced to problem (9) see [26].

4.1. Von Neumann's algorithm

In 1948, Von Neumann proposed to Dantzig an algorithm for finding a feasible linear programming solution, which was first published in the early 90's [27,28]. This algorithm has interesting properties,

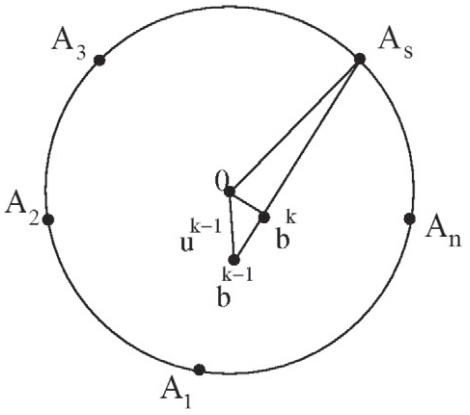

Fig. 1. Illustration of Von Neumann's algorithm.

such as simplicity and fast initial convergence. However, it is not very practical for solving linear problems, since its convergence is very slow.

First, the algorithm finds the column A_s of A which forms the largest angle with the residual $b^{k-1} = Ax^{k-1}$ and then the next residual b^k is given by the projection of the origin on the line segment connecting b^{k-1} to A_s. See Fig. 1.

In Von Neumann's algorithm, the new residual is smaller than the previous one, as can easily be seen in Fig. 1, the triangle $0b^{k-1}b^k$ has hypotenuse $u^{k-1} = 0b^{k-1}$ and side $u^k = 0b^k$.

The stopping criterion used in Von Neumann's algorithm in [13] is the relative error between $||b^{k-1}||$ and $||b^k||$, i.e., $(||b^{k-1} - b^k||)/||b^{k-1}||$, which must be smaller than a certain specified percentage.

Moreover, the effort per iteration of the Von Neumann algorithm is dominated by matrix vector multiplication needed in the selection of column A_s which is $O(nz(A))$ where $nz(A)$ is the number of the entries of A. For very sparse matrices this is an affordable cost.

4.2. Optimal pair adjustment algorithm

In his Ph.D. thesis [26], Gonçalves studied Von Neumann's algorithm and introduced four new algorithms based on it. Among them, emphasis is given to the optimal pair adjustment algorithm, the one that better performed in practice.

The weight-reduction algorithm was first proposed as an attempt to improve the efficiency of Von Neumann's algorithm. It is based on the idea that the residual b^{k-1} can be moved closer to the origin 0, increasing the weight x_j for a given column A_j and decreasing the weight x_i for another column A_i.

In particular, it is expected that the new residual b^k is closer to the origin 0 that the residual b^{k-1} if the weight in column A_{s^+} is increased when A_{s^+} has the largest angle with the residual b^{k-1} and the weight in column A_{s^-} is decreased when A_{s^-} has the smallest angle with the residual b^{k-1}. This corresponds to moving the residual b^{k-1} towards $A_{s^+} - A_{s^-}$. The new residual b^k is the point that minimizes the distance from the origin 0 to this line. Notice that this distance is minimized subject to the maximum possible decrease of x_{s^-}. Since $x_j \geqslant 0$ for all j, then x_{s^-} can be decreased until it vanishes.

The optimal pair adjustment algorithm is a generalization of the weight-reduction algorithm designed to give the maximum possible freedom to two of the weights x_j (see [13]). In a way, we can say that it prioritises two variables for each iteration, because it finds the optimal value for two coordinates and adjusts the remaining coordinates according to these values.

Similar to the weight-reduction algorithm, the optimal pair adjustment algorithm starts by identifying the vectors A_{s^+} and A_{s^-}, which have the largest and smallest angle with b^{k-1}, respectively. Then the values $x^k_{s^+}$, $x^k_{s^-}$ and λ are found, where $x^k_j = \lambda x^{k-1}_j$ for all $j \neq s^+$ and $j \neq s^-$, that minimize the distance from b^k and the origin while satisfying the convexity and the non-negativity constraints. The solution of this optimization problem is easily computed examining the KKT conditions. The main difference between the weight-reduction algorithm and the optimal pair adjustment algorithm is that only the weights of A_{s^+} and A_{s^-} are changed in the first algorithm while in the second algorithm all other weights are changed.

The optimal pair adjustment algorithm inherits the best properties of the algorithm of Von Neumann. The work per iteration of the optimal pair adjustment algorithm is of the same order as the work per iteration of the von Neumann algorithm. Although Gonçalves has proved in [13] that, in terms of the convergence this algorithm is faster than Von Neumann's, it is impractical to solve linear programming problems, as its convergence is very slow.

Details of the optimal pair adjustment algorithm are given in [26].

4.3. Optimal adjustment algorithm for p coordinates

The optimal pair adjustment algorithm prioritizes two coordinates for each iteration. This algorithm will be referred to as a 2 variable algorithm. Using the same idea of the 2 variable algorithm we can generalize it and build the algorithm for p variables, where p is limited by the problem dimension. For this, instead of only two columns to be used to formulate the problem, any number of columns can be used and then, importance will be given for any desired subset of variables.

Therefore, the optimal adjustment algorithm for p coordinates is developed generalizing the ideas in [13] for the optimal pair adjustment algorithm, The major advantage of this algorithm is to find better directions and still maintain their simplicity, i.e., in each iteration only matrix vector multiplication is performed and a positive definite linear system of small dimension, in comparison with the problem size, is solved.

The strategy of prioritizing variables is free and can be chosen according to the problem that will be solved. A natural choice is taking $p/2$ columns forming the largest angle with the vector b^k and the remaining $p/2$ columns that make the smallest angle with vector b^k. If p is odd then one more column is added to the set of vectors that form the largest angle with the vector b^k for example.

The structure of the optimal adjustment algorithm for p coordinates is similar to the optimal pair adjustment algorithm. It begins by identifying the s_1 columns that make the largest angle with the vector b^{k-1}, then s_2 columns that make the smallest angle with the vector b^{k-1} are determined, where $s_1 + s_2 = p$ and p is the number of columns that will be prioritized. Next, the optimization subproblem is solved and the residual and current points are updated.

4.4. Subproblem solution using interior point methods

For each iteration of the optimal adjustment algorithm for p coordinates, it is necessary to solve the subproblem (10). This subproblem is solved finding a solution that satisfies a linear equation system of the order at most $(p + 1)$. A way for this is to verify all possible cases of the feasible solutions of the system. The drawback that comes naturally from solving the subproblem in this way is that the number of possible cases to be verified grows exponentially with the value of p as shown next.

To solve the subproblem (10) first the variable λ_0 must be eliminated. To do this, the equality constraint is rewritten as:

$$\lambda_0 = \frac{1 - \sum_{i=1}^{s_1} \lambda_{\eta_i^+} - \sum_{j=1}^{s_2} \lambda_{\eta_j^-}}{1 - \sum_{i=1}^{s_1} x^{k-1}_{\eta_i^+} - \sum_{j=1}^{s_2} x^{k-1}_{\eta_j^-}}$$

Substituting this expression where appropriate, the problem reduces to

Algorithm. Optimal adjustment algorithm for p coordinates.

Given: $x^0 \geqslant 0$, with $e^T x^0 = 1$. Compute $b^0 = Ax^0$.
For $k = 1, 2, 3, \ldots$ do

(1) Compute:

$$\{A_{\eta_1^+}, \ldots, A_{\eta_{s_1}^+}\} \text{ which make the largest angle with the vector } b^{k-1}.$$
$$\{A_{\eta_1^-}, \ldots, A_{\eta_{s_2}^-}\} \text{ which make the smallest angle with the vector } b^{k-1}$$
$$\text{and such that } x_i^{k-1} > 0, i = \eta_1^-, \ldots, \eta_{s_2}^-, \text{ where } s_1 + s_2 = p.$$
$$v_{k-1} = \text{minimum}_{i=1,\ldots,s_1} \left\{ A_{\eta_i^+}^t b^{k-1} \right\}.$$

(2) If $v_{k-1} > 0$, then **STOP**. Problem (9) is infeasible.
(3) Solve the subproblem

$$\begin{aligned} &\text{Min } ||\bar{b}||^2 \\ &\text{s.t. } \lambda_0 \left(1 - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1} \right) + \sum_{i=1}^{s_1} \lambda_{\eta_i^+} + \sum_{j=1}^{s_2} \lambda_{\eta_j^-} = 1, \\ &\quad \lambda_{\eta_i^+} \geqslant 0, \quad \text{for } i = 1, \ldots, s_1, \\ &\quad \lambda_{\eta_i^-} \geqslant 0, \quad \text{for } j = 1, \ldots, s_2. \end{aligned} \tag{10}$$

where $\bar{b} = \lambda_0 \left(b^{k-1} - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} A_{\eta_i^+} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1} A_{\eta_j^-} \right) + \sum_{i=1}^{s_1} \lambda_{\eta_i^+} A_{\eta_i^+} + \sum_{j=1}^{s_2} \lambda_{\eta_j^-} A_{\eta_j^-}$.

(4) Update:

$$b^k = \lambda_0 \left(b^{k-1} - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} A_{\eta_i^+} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1} A_{\eta_j^-} \right) + \sum_{i=1}^{s_1} \lambda_{\eta_i^+} A_{\eta_i^+} + \sum_{j=1}^{s_2} \lambda_{\eta_j^-} A_{\eta_j^-},$$
$$u^k = ||b^k||,$$
$$x_j^k = \begin{cases} \lambda_0 x_j^{k-1}, \; j \notin \left\{ \eta_1^+, \ldots, \eta_{s_1}^+, \eta_1^-, \ldots, \eta_{s_2}^- \right\}, \\ \lambda_{\eta_i^+}, \quad j = \eta_i^+, i = 1, \ldots, s_1, \\ \lambda_{\eta_j^-}, \quad j = \eta_j^-, j = 1, \ldots, s_2. \end{cases}$$
$$k = k + 1.$$

$$\begin{aligned} &\text{Min} \quad ||\bar{b}||^2 \\ &\text{s.t.} \quad 1 - \sum_{i=1}^{s_1} \lambda_{\eta_i^+} - \sum_{j=1}^{s_2} \lambda_{\eta_j^-} \geqslant 0, \\ &\quad \lambda_{\eta_i^+} \geqslant 0, \quad \text{for } i = 1, \ldots, s_1, \\ &\quad \lambda_{\eta_i^-} \geqslant 0, \quad \text{for } j = 1, \ldots, s_2, \end{aligned} \tag{11}$$

where,

$$\bar{b} = \lambda_0 \left(b^{k-1} - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} A_{\eta_i^+} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1} A_{\eta_j^-} \right) + \sum_{i=1}^{s_1} \lambda_{\eta_i^+} A_{\eta_i^+} + \sum_{j=1}^{s_2} \lambda_{\eta_j^-} A_{\eta_j^-}.$$

Defining:
$$g_0(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) = \sum_{i=1}^{s_1} \lambda_{\eta_i^+} + \sum_{j=1}^{s_2} \lambda_{\eta_j^-} - 1,$$
$$g_i(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) = -\lambda_{\eta_i^+}, \quad i = 1, \ldots, s_1,$$
$$h_j(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) = -\lambda_{\eta_j^-}, \quad j = 1, \ldots, s_2$$

and denoting the objective function by $f(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-})$ then the KKT conditions of this subproblem are the following:

$$\begin{cases} \nabla f(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) + \sum_{i=0}^{s_1} \mu_{g_i} \nabla g_i(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) \\ \quad + \sum_{j=1}^{s_2} \mu_{h_i} \nabla h_i(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) = 0, \\ \mu_{g_i} g_i(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) = 0, \ \text{for } i = 0, \ldots, s_1, \\ \mu_{h_j} h_j(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) = 0, \ \text{for } j = 1, \ldots, s_2, \\ \mu_{g_i} \geqslant 0, \ \text{for } i = 0, \ldots, s_1, \\ \mu_{h_j} \geqslant 0, \ \text{for } j = 1, \ldots, s_2, \\ g_i(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) \leqslant 0, \ \text{for } i = 0, \ldots, s_1, \\ h_j(\lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}) \leqslant 0, \ \text{for } j = 1, \ldots, s_2. \end{cases} \tag{12}$$

Then the subproblem (11) is solved by selecting a feasible solution among all the possibilities that satisfy the KKT conditions (12). For this, all cases of possible values for the variables $\lambda_{\eta_i^+}, i = 1, \ldots, s_1$ and $\lambda_{\eta_j^+}, j = 1, \ldots, s_2$ must be analyzed. The total number of cases to be analyzed when p coordinates are modified is:

$$1 + 2C_1^p + 2C_2^p + 2C_3^p + \cdots + 2C_p^p = 2^{(p+1)} - 1$$

This strategy is inefficient even though values of p are not too large. In order to deal with this difficulty, subproblem (10) is approached differently and solved using interior point methods. A great advantage of using interior point methods to solve the subproblem (10), is that the computational cost to solve a large scale problem with a matrix of small order is insignificant. Moreover, its implementation is easier to perform.

To apply an interior point method, the subproblem is rewritten in matrix form:

$$\begin{aligned} \text{Min} \quad & \tfrac{1}{2} ||W\lambda||^2 \\ \text{s.t.} \quad & a^T \lambda = 1, \\ & \lambda \geqslant 0, \end{aligned} \tag{13}$$

where

$$W = \left[\overline{w} \; A_{\eta_1^+} \ldots A_{\eta_{s_1}^+} A_{\eta_1^-} \ldots A_{\eta_{s_2}^-} \right],$$
$$\overline{w} = b^{k-1} - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} A_{\eta_i^+} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1} A_{\eta_j^-}, \tag{14}$$

$$\lambda = \left(\lambda_0, \lambda_{\eta_1^+}, \ldots, \lambda_{\eta_{s_1}^+}, \ldots, \lambda_{\eta_1^-}, \ldots, \lambda_{\eta_{s_2}^-}\right),$$
$$a = (a_1, 1, \ldots, 1),\ a_1 = 1 - \sum_{i=1}^{s_1} x_{\eta_i^+}^{k-1} - \sum_{j=1}^{s_2} x_{\eta_j^-}^{k-1}.$$

The KKT equations of problem (13) are given by:

$$\begin{aligned} &W^T W\lambda + a\gamma - \mu = 0,\\ &\mu^T\lambda = 0,\\ &a^T\lambda - 1 = 0,\\ &-\lambda \leqslant 0. \end{aligned} \tag{15}$$

where γ is free and $\mu \geqslant 0$, the variables γ and μ are Lagrange multipliers of equality and inequality constraints respectively and $(p+1)\times(p+1)$ is the dimension of the matrix $W^T W$.

The interior point method is applied to QP problems.

The linear system arising at each iteration of the interior point method applied to (15) has the following form:

$$\begin{bmatrix} W^T W & a & -I \\ U & 0 & \Lambda \\ a^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta\lambda \\ \delta\gamma \\ \delta\mu \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \tag{16}$$

where
$$\begin{aligned} U &= diag(\mu),\\ \Lambda &= diag(\lambda),\\ r_1 &= \mu - a\gamma - W^T W\lambda,\\ r_2 &= -\gamma\lambda,\\ r_3 &= 1 - a^T\lambda. \end{aligned}$$

By performing some simple variable eliminations, we obtain that the directions $\delta\mu$, $\delta\lambda$ and $\delta\gamma$ are given by:

$$\begin{aligned} &\delta\mu = \Lambda^{-1}r_2 - \Lambda^{-1}Ud\lambda,\\ &\delta\lambda = (W^T W + \Lambda^{-1}U)^{-1}r_4 - (W^T W + \Lambda^{-1}U)^{-1}a\delta\gamma,\\ &a^T(W^T W + \Lambda^{-1}U)^{-1}a\delta\gamma = a^T(W^T W + \Lambda^{-1}U)^{-1}r_4 - r_3, \end{aligned} \tag{17}$$

where $r_4 = r_1 + \Lambda^{-1}r_2$.

Thus, to compute the directions, the following linear systems must be solved:

$$\begin{aligned} &(W^T W + \Lambda^{-1}U)v_1 = a\\ &(W^T W + \Lambda^{-1}U)v_2 = r_4 \end{aligned} \tag{18}$$

Note that this is a positive definite matrix of order $p+1$ and both systems can be solved using the same factorization.

In [14], it was proved that the performance of the new method is better than the Von Neumann's algorithm. Furthermore, it was shown that if $p_2 \geqslant p_1$, then the optimal adjustment algorithm for p_2 coordinates has a better performance than the optimal adjustment algorithm for p_1 coordinates.

On the other hand, it is not advisable to chose a very large value of p since there is a cost of building and updating matrix W in (18). Such cost is negligible for small values of p. However, it becomes noticeable for larger values.

Table 1
Problem statistics.

Problem	Rows	Columns	Collection	Problem	Rows	Columns	Collection
25fv47	788	1843	NETLIB	maros	655	1437	NETLIB
adlittle	55	137	NETLIB	stocfor2	1980	2868	NETLIB
agg2	514	750	NETLIB	els19	4350	13186	QAPLIB
agg3	514	750	NETLIB	chr25a	8149	15325	QAPLIB
bandm	240	395	NETLIB	chr22b	5587	10417	QAPLIB
blend	71	111	NETLIB	nug05	210	225	QAPLIB
bnl2	1964	4008	NETLIB	nug05-3rd	1410	825	QAPLIB
boeing1	331	697	NETLIB	nug06	372	486	QAPLIB
boeing2	125	264	NETLIB	nug07	602	931	QAPLIB
bore3d	81	138	NETLIB	nug08	912	1632	QAPLIB
capri	241	436	NETLIB	nug12	3192	8856	QAPLIB
d6cube	403	5444	NETLIB	nug12m	3192	8856	QAPLIB
degen2	444	757	NETLIB	nug15	6330	22275	QAPLIB
degen3	1503	2604	NETLIB	nug15m	6330	22275	QAPLIB
e226	198	429	NETLIB	qap12	2794	8856	QAPLIB
etamacro	334	669	NETLIB	qap15	5698	22275	QAPLIB
finnis	438	935	NETLIB	rou20	7359	37640	QAPLIB
forplan	121	447	NETLIB	scr15	2234	6210	QAPLIB
israel	174	316	NETLIB	scr20	5079	15980	QAPLIB
kb2	43	68	NETLIB	ste36b	27683	131076	QAPLIB

5. Computational experiments

The main objective of these computational experiments is to compare the performance of a version of PCx code [6] that has incorporated a hybrid approach of preconditioning with a new version of the same code in which was added in the optimal adjustment algorithm for p coordinates to be used in the phase of exchange of preconditioners.

Both versions of the PCx were coded in C except the controlled Cholesky factorization, which was implemented in FORTRAN. The gcc and gfortran compiler were used. All the experiments were performed on an Intel Core 2 Duo 64 bits T7250, 2GB RAM and 2.2GHz with operating system Linux.

5.1. Test problems

In this work, 40 test problems were considered, all they are freely available. The problems are from the NETLIB collection and the QAPLIB test set. Most of the chosen test problems are the same as the ones adopted in [8], where the hybrid approach performed better than the direct approach. The main reason for such result is the large number of nonzero entries in the ADA^T Cholesky factor.

Table 1 presents the dimensions of the test problems after preprocessing.

5.2. Parameters

For the experiments the PCx's default parameters were adopted except the multiple centrality corrections [2], which were disabled. Other important parameters used by the hybrid approach of preconditioning and the optimal adjustment algorithm for p coordinates were determined as described below.

5.2.1. Choice of p

For the optimal adjustment algorithm for p coordinates to work properly an appropriate choice of the parameter p is essential. Thus, several computational experiments were done to determine a heuristic that works well in any linear programming problem. With results obtained in the tests it became clear that the value of p must be chosen depending on the size of the problem. Recalling that m is the number of rows and n is the number of columns of the linear problem constraint matrix, the heuristic that have shown better results was as follows:

$$
\begin{aligned}
0 &< (m+n) \leqslant 10000 &&\longrightarrow p = 4;\\
10000 &< (m+n) \leqslant 20000 &&\longrightarrow p = 8;\\
20000 &< (m+n) \leqslant 400000 &&\longrightarrow p = 20;\\
400000 &< (m+n) \leqslant 600000 &&\longrightarrow p = 40;\\
600000 &< (m+n) &&\longrightarrow p = 80.
\end{aligned}
$$

To observe the performance of the PCxMod when the optimal adjustment algorithm for p coordinates and the optimal pair adjustment algorithm are used, the same set of test problems was solved. First $p = 2$ (2-coord) was used, which represents the optimal pair adjustment algorithm and then the value of p conform the criterion previously described was used.

The results obtained are presented and commented in Section 5.3.

5.2.2. Stopping criterion

The number of iterations of the optimal adjustment algorithm for p coordinates to be performed is an important parameter to be determined, since it directly influences the performance of the PCx. This algorithm achieves better results when the solution is determined with good accuracy. However, in some cases, the number of iterations needed for convergence of the algorithm can be very large, making it impractical to use. In these cases, a maximum number of iterations should be adopted. So the stopping criterion for the optimal adjustment algorithm for p coordinates used is the following: maximum number of iterations (100) or the relative error of the residual norm smaller than a given tolerance (10^{-4}) (The one that occurs first).

The preconditioned conjugate gradient method is used with a termination criteria set by the Euclidean residual norm $||.||_2$. For solving both systems *(affine direction* and *final direction)*, the termination criteria is set as $||r_k|| < 10^{-4}$. When the optimality gap is less than 10^{-5} or change of phases is detected, the criteria change to $||r_k|| < 10^{-8}$. The maximum number of iterations of the conjugate gradient method is equal to the system dimension.

5.3. Results

In Table 2 are compared the total number of interior point iterations (column: Iterations) and the total running time in seconds (column: Time) of the two versions of the PCx which a hybrid preconditioner approach. PCxMod-p is the version that use the optimal adjustment algorithm for p coordinates in the exchange of preconditioners and PCxMod is the one that does not adopt it. Column: p shows the value of p used by the optimal adjustment algorithm. Column: ItAux gives the number of iterations performed by the optimal adjustment algorithm for p coordinates. The values in bold face on Tables 2 and 3 mean the smallest iteration count or running time between the two approaches.

The PCxMod-p takes less time to obtain the optimal solution in 82.5% of the tested problems. The total number of iterations was reduced in 15% of the cases.

Although the total number of iterations did not decreased in several problems using of the optimal adjustment algorithm for p coordinates, the improved point reduced the running time after changing the preconditioner showing that it delivers a better point.

Analyzing the results on Table 2, it can be seen that there are problems such as 25fv47, agg3, maros, ch25a, chr22b, nug05-3rd, qap12, among others where the number of iterations was greater than or equal for the version PCxMod-p but the total running time was smaller. This happens because the number of iterations of the conjugate gradient performed during the resolution of these problems is smaller. In other words, using the new improved point given by optimal adjustment algorithm for p coordinates in the exchange of preconditioners produces linear systems easier to solve. For example, the problem chr25a, PCxMod performed 31657 iterations of the conjugate gradient method while PCxMod-p performed 27676 iterations, a reduction of approximately 12.5% in the number of iterations of conjugate gradient and, consequently, reduced 19.40% of the total running time.

It should be mentioned that the required total time to obtain a solution for the optimal adjustment algorithm for p coordinates is not significant in relation to the resolution total time of the problems. This time is almost null in many of the tested problems. Considering the largest problem (ste36b) the

Table 2
PCxMod-p × PCxMod.

Problem	p	ItAux	Iterations		Time	
			PCxMod-p	PCxMod	PCxMod-p	PCxMod
25fv47	4	10	26	26	**2.40**	2.46
adlittle	4	10	12	12	**0.00**	0.01
agg2	4	4	24	24	0.67	**0.65**
agg3	4	4	22	22	**0.48**	0.50
bandm	4	5	17	17	0.15	0.15
blend	4	10	10	10	**0.00**	0.01
bnl2	4	10	**36**	37	**6.83**	7.36
boeing1	4	10	20	20	**0.30**	0.31
boeing2	4	10	14	14	**0.03**	0.04
bore3d	4	4	16	16	0.02	0.02
capri	4	2	19	19	0.09	0.09
d6cube	4	2	19	19	**2.02**	2.06
degen2	4	3	12	12	**0.32**	0.33
degen3	4	14	16	16	**6.59**	6.95
e226	4	10	18	18	0.16	**0.15**
etamacro	4	6	**26**	27	**0.21**	0.23
finnis	4	4	25	25	**0.21**	0.23
forplan	4	10	**23**	24	**0.17**	0.19
israel	4	10	22	**21**	0.13	0.13
kb2	2	10	**12**	13	**0.00**	0.01
maros	4	11	20	20	**1.97**	2.04
stocfor2	4	3	21	21	**1.68**	1.70
els19	20	11	32	**31**	**100.19**	106.93
chr25a	8	4	29	**28**	**40.75**	50.55
chr22b	8	4	29	29	**15.38**	16.56
nug05	4	4	6	6	0.03	0.03
nug05-3rd	2	4	6	6	**1.35**	1.60
nug06	2	2	6	6	**0.11**	0.13
nug07	4	11	**10**	11	**0.47**	0.50
nug08	4	4	9	9	**1.28**	1.37
nug12	8	4	20	20	**155.47**	175.08
nug12m	8	3	20	20	**157.50**	173.52
nug15	8	4	23	23	**1958.65**	2181.04
nug15m	8	4	24	**23**	**1963.81**	2184.58
qap12	8	3	20	20	**157.13**	178.99
qap15	20	4	**23**	*	**2155.71**	*
rou20	20	3	24	24	**1908.54**	2029.99
scr15	4	3	24	24	**10.23**	11.64
scr20	20	4	21	21	**144.35**	147.70
ste36b	40	3	35	37	**25682.43**	26586.91

* Means that the method failed.

time spent by the optimal adjustment algorithm is of 1.71 seconds, which represents less than 0.01 % of the total running time.

The same set of test problems was solved using only the version PCxMod-p, first for $p = 2$ (2-coord), which represents the optimal pair adjustment algorithm and then for the value of p determined as the criterion previously described. The results are presented in Table 3.

The performance of the algorithm is improved by increasing the value of p. This happens because the residual b^k has a greater reduction from one iteration to another. Moreover, the point given by the algorithm as a solution is different for each value âŁ‹âŁ‹of p because of the stopping criterion used. Therefore, the points for values greater than $p = 2$ are better in most cases.

According to results the total running time was lower in about 73% of the problems for p-coordinates. For $p = 2$ such number is approximately 12%. The number of iterations was equal or close in both cases for most problems.

It should be mentioned that some of the problems were only solved when $p > 2$ coordinates were used. This confirms the importance of choosing an appropriate parameter p.

Table 3
p-Coordinates × 2-coordinates.

Problem	*p*	Iterations		Time	
		p-coord	2-coord	p-coord	2-coord
25fv47	4	26	26	**2.40**	2.42
adlittle	4	12	12	0.00	0.00
agg2	4	24	24	**0.67**	0.68
agg3	4	22	22	**0.48**	0.49
bandm	4	17	17	**0.15**	0.16
blend	4	10	10	**0.00**	0.01
bnl2	4	36	37	**6.83**	7.48
boeing1	4	20	20	**0.30**	0.35
boeing2	4	**14**	15	0.03	0.03
bore3d	4	16	16	0.02	0.02
capri	4	19	19	0.09	0.09
d6cube	4	19	19	**2.02**	2.04
degen2	4	12	12	**0.32**	0.35
degen3	4	**16**	19	**6.59**	7.48
e226	4	**18**	19	0.16	0.16
etamacro	4	**26**	27	**0.21**	0.23
finnis	4	**25**	26	**0.21**	0.25
forplan	4	**23**	24	**0.17**	0.18
israel	4	**22**	24	0.13	0.13
kb2	4	12	12	0.00	0.00
maros	4	20	20	**1.97**	2.02
stocfor2	4	21	21	**1.68**	1.71
els19	20	32	**31**	**100.19**	106.98
chr25a	8	29	29	**40.75**	46.26
chr22b	8	29	29	**15.38**	15.39
nug05	4	6	6	**0.03**	*
nug05-3rd	4	6	6	1.35	1.35
nug06	4	6	6	0.11	0.11
nug07	4	**10**	11	**0.47**	0.48
nug08	4	**9**	10	**1.28**	1.46
nug12	8	20	20	**155.47**	171.07
nug12m	8	20	20	**157.50**	162.97
nug15	8	23	23	**1958.65**	*
nug15m	8	24	**23**	**1963.81**	*
qap12	8	20	20	157.13	**155.42**
qap15	20	**23**	*	**2155.71**	*
rou20	20	24	24	**1908.54**	1964.26
scr15	4	24	24	**10.23**	10.26
scr20	20	**21**	22	**144.35**	157.25
ste36b	40	**35**	37	**25682.43**	26586.91

* Means that the method failed.

The computational results demonstrated that even with a small number of iterations, the optimal adjustment algorithm for *p* coordinates is an important tool when applied in combination with the hybrid preconditioner approach, bringing more speed and improving the robustness of it.

6. Conclusion and future work

In this work, the optimal adjustment algorithm for *p* coordinates, whose main advantages are simplicity and fast initial convergence, was used to accelerate the convergence of the interior point methods.

A hybrid preconditioner approach combines the optimal adjustment algorithm for *p* coordinates with interior point methods in the change of phases between preconditioners.

The controlled Cholesky factorization (CCF) preconditioner is used in the initial interior point iterations and the splitting preconditioner is applied in the final iterations, where the linear systems are highly ill conditioned. The splitting preconditioner works very well in the final iterations. However,

it is not effective in the first ones. On the other hand, the CCF preconditioner cannot deal with the systems close to a solution leaving a gap between both algorithms that sometimes cannot be closed by this approach and it fails.

The optimal adjustment algorithm for p coordinates is efficient in closing the gap giving more speed in the solution of at most all solved problems and allowing the hybrid approach to solve some problems to optimality, which could not be obtained otherwise. It delivers a point close to the solution after the controlled Cholesky factorization is applied, allowing the splitting preconditioner to start with a more favorable point. That is, using the optimal adjustment algorithm for p coordinates in exchange for phase leading to a more robust implementation.

With respect to future research, the performance of this approach can be improved by developing more sophisticated heuristics for the choice of the number of coordinates and for changing the phase. New strategies for choosing the p columns should also be investigated.

Acknowledgement

This research was partially sponsored by the Brazilian Council for the Development of Science and Technology (CNPq).

References

[1] M. Colombo, J. Gondzio, Further development of multiple centrality correctors for interior point methods, Comput. Optim. Appl. 41 (2008) 277–305.
[2] J. Gondzio, Multiple centrality corrections in a primal–dual method for linear programming, Comput. Optim. Appl. 6 (1996) 137–156.
[3] I.J. Lustig, R.E. Marsten, D.F. Shanno, On implementing Mehrotra's predictor–corrector interior point method for linear programming, SIAM J. Optim. 2 (1992) 435–449.
[4] S. Mehrotra, On the implementation of a primal–dual interior point method, SIAM J. Optim. 2 (1992) 575–601.
[5] I. Adler, M.G.C. Resende, G. Veiga, N. Karmarkar, An implementation of Karmarkar's algorithm for linear programming, Math. Program. 44 (1989) 297–335.
[6] J. Czyzyk, S. Mehrotra, M. Wagner, S.J. Wright, PCx an interior point code for linear programming, Optim. Methods Softw. 11 (2) (1999) 397–430.
[7] L. Bergamaschi, J. Gondzio, M. Venturin, G. Zilli, Inexact constraint preconditioners for linear systems arising in interior point methods, Comput. Optim. Appl. 36 (2007) 137–147.
[8] S. Bocanegra, F.F. Campos, A.R.L. Oliveira, Using a hybrid preconditioner for solving large-scale linear systems arising from interior point methods, Comput. Optim. Appl. 36 (2007) 149–164.
[9] J.S. Chai, K.C. Toh, Preconditioning and iterative solution of symmetric indefinite linear systems arising from interior point methods for linear programming, Comput. Optim. Appl. 36 (2007) 221–247.
[10] A.R.L. Oliveira, D.C. Sorensen, A new class of preconditioners for large-scale linear systems from interior point methods for linear programming, Linear Algebra Appl. 394 (2005) 1–24.
[11] F.F. Campos, N.R.C. Birkett, An efficient solver for multi-right hand side linear systems based on the CCCG (η) method with applications to implicit time-dependent partial differential equations, SIAM J. Sci. Comput. 19 (1998) 126–138.
[12] M.I. Velazco, A.R.L. Oliveira, F.F. Campos, A note on hybrid preconditions for large scale normal equations arising from interior–point methods, Optim. Methods Softw. 25 (2010) 321–332.
[13] J.P.M. Gonçalves, R.H. Storer, J. Gondzio, A family of linear programming algorithms based on an algorithm by von Neumann, Optim. Methods Softw. 24 (2009) 461–478.
[14] J. Silva, Uma Família de Algoritmos para Programação Linear Baseada no Algoritmo de Von Neumann, Ph.D. thesis, IMECC – UNICAMP, Campinas, SP, 2009 (in portuguese).
[15] C. Durazzi, V. Ruggiero, Indefinitely preconditioned conjugate gradient method for large sparse equality and inequality constrained quadratic problems, Numer. Linear Algebra Appl. 10 (2003) 673–688.
[16] C. Keller, N.I.M. Gould, A.J. Wathen, Constraint preconditioning for indefinite linear systems, SIAM J. Matrix Anal. Appl. 21 (2000) 1300–1317.
[17] A. Altman, J. Gondzio, Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization, Optim. Methods Softw. 11 (1999) 275–302.
[18] L. Bergamaschi, J. Gondzio, G. Zilli, Preconditioning indefinite systems in interior point methods for optimization, Comput. Optim. Appl. 28 (2004) 149–171.
[19] F.F. Campos, Analysis of conjugate gradients – type methods for solving linear equations, Ph.D. thesis, Oxford University Computing Laboratory, Oxford, 1995.
[20] M.G.C. Resende, G. Veiga, An efficient implementation of a network interior point method, DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 12 (1993) 299–348.
[21] A.R.L. Oliveira, A new class of preconditioners for large-scale linear systems from interior point methods for linear programming, Technical Report, Ph.D. Thesis, TR97-11, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1997.

[22] A. George, E. Ng, An implementation of Gaussian elimination with partial pivoting for sparse systems, SIAM J. Sci. Statist. Comput. 6 (1985) 390–409.
[23] I.S. Duff, G.A. Meurant, The effect of ordering on preconditioned conjugate gradients, BIT 29 (1989) 635–657.
[24] G.E. Forsythe, E.G. Straus, On best conditioned matrices, Proc. Amer. Math. Soc. 6 (1955) 340–345.
[25] M.T. Jones, P.E. Plassmann, An improved incomplete Cholesky factorization, ACM Trans. Math. Softw. 21 (1995) 5–17.
[26] J.P.M. Gonçalves, A family of linear programming algorithms based on the Von Neumann algorithm, Ph.D. thesis, Lehigh University, Bethlehem, 2004.
[27] G.B. Dantzig, Converting a converging algorithm into a polynomially bounded algorithm, Technical Report, Stanford University, 1991.
[28] G.B. Dantzig, An ϵ-precise feasible solution to a linear program with a convexity constraint in $\frac{1}{\epsilon^2}$ iterations independent of problem size, Technical Report, Stanford University, 1992.